

偏移量，后12位

二级页表，中间10位

一级页表，高10位

页式管理

在保护模式中，x86 体系结构将内存地址分成三种：逻辑地址（也称虚地址）、线性地址和物理地址。逻辑地址即是程序指令中使用的地址，物理地址是实际访问内存的地址。逻辑地址通过段式管理的地址映射可以得到线性地址，线性地址通过页式管理的地址映射得到物理地址。

该实验将逻辑地址不加转换直接映射成线性地址，所以我们在下面的讨论中可以对这两个地址不加区分（目前的 OS 实现也是不加区分的）



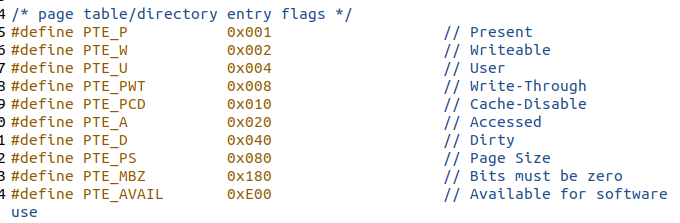
PDT(页目录表),PDE(页目录项),PTT(页表),PTE(页表项)之间的关系:页表保存页表项，页表项被映射到物理内存地址；页目录表保存页目录项，页目录项映射到页表。

请描述页目录项（Page Directory Entry）和页表项（Page Table Entry）中每个组成部分的含义以及对ucore而言的潜在用处。

PDE：一级页表

PTE：二级页表

PDE和PTE都是4B大小的一个元素，其高20bit被用于保存索引，低12bit用于保存属性，但是由于用处不同，内部具有细小差异，如图所示：





**页目录项（PDE）**具体位的功能：

bit 0§: Present，用来确认对应的页表是否存在。

bit 1(R/W): read/write，若该位为0 ,则只读，否则可写。

bit 2(U/S): user/supervisor，用来确认用户态下是否可以访问。

bit 3(PWT): page-level write-through，表示是否使用write through缓存写策略。

bit 4(PCD): page-level cache disable，表示是否不对该页进行缓存。

bit 5(A): accessed，用来确认对应页表是否被访问过。

bit 6: 忽略。

bit 7(PS): Page size，这个位用来确定32位分页的页大小，当该位为1且CR4的PSE位为1时，页大小为4M，否则为4K。

bit 11:8: 忽略。

bit 32:12: 页表的PPN（页对齐的物理地址）。

页表项除了第7，8位与PDE不同，其余位作用均相同。

bit 7(PAT): 如果支持PAT分页，间接决定这项访问的4K页的内存类型;如果不支持，这位保留（必须为 0 ）。

bit 8(G): global位。当CR4的PGE位为1时，若该位为1，翻译是全局的;否则，忽略该位。

其中被忽略的位可以被操作系统用于实现各种功能；和权限相关的位可以用来增强ucore的内存保护机制；access位可以用来实现内存页交换算法。

**页表项（PTE）**除了第7，8位与PDE不同，其余位作用均相同。

bit 7(PAT): 如果支持PAT分页，间接决定这项访问的4K页的内存类型;如果不支持，这位保留（必须为 0 ）。

bit 8(G): global位。当CR4的PGE位为1时，若该位为1，翻译是全局的;否则，忽略该位。

其中被忽略的位可以被操作系统用于实现各种功能；和权限相关的位可以用来增强ucore的内存保护机制；access位可以用来实现内存页交换算法。

如果ucore执行过程中访问内存，出现了页访问异常，请问硬件要做哪些事情？

当启动分页机制以后，如果一条指令或数据的虚拟地址所对应的物理页不在内存中或者访问的类型有误（比如写一个只读页或用户态程序访问内核态的数据等），就会发生页错误异常。

而产生页面异常的原因主要有:

①目标页面不存在（页表项全为0，即该线性地址与物理地址尚未建立映射或者已经撤销）；

②相应的物理页面不在内存中（页表项非空，但Present标志位=0，比如将页表交换到磁盘）；

③访问权限不符合（比如企图写只读页面）。

当出现上面情况之一,那么就会产生页面page fault(#PF)异常。产生异常的虚拟地址存储在CR2中，并且将是page fault的错误类型保存在error code中，之后需要往中断时的栈中压入EFLAGS,CS,EIP,ERROR CODE，如果这页访问异常很不巧发生在用户态，还需要先压入SS,ESP并切换到内核态。引发异常后将外存的数据换到内存中，进行上下文切换，退出中断，返回到中断前的状态。

这里说一下控制寄存器CR0-4的作用：

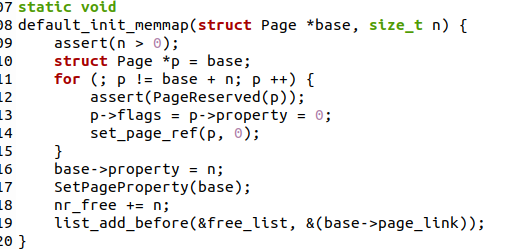
·CR0的0位是PE位，如果为1则启动保护模式，其余位也有自己的作用

·CR1是未定义控制寄存器，留着以后用

·CR2是页故障线性地址寄存器，保存最后一次出现页故障的全32位线性地址

·CR3是页目录基址寄存器，保存PDT的物理地址

·CR4在Pentium系列处理器中才实现，它处理的事务包括诸如何时启用虚拟8086模式等。



主要缺陷在于遍历中间的空闲页没有被加入到空闲页表中

另一个缺陷在于中间页的 flags 处未被设为 PG\_property(其值为 1)，使用的函数是SetPageProperty

#define SetPageProperty(page) set\_bit(PG\_property, &((page)->flags))